

# Algorithmique et structures de données

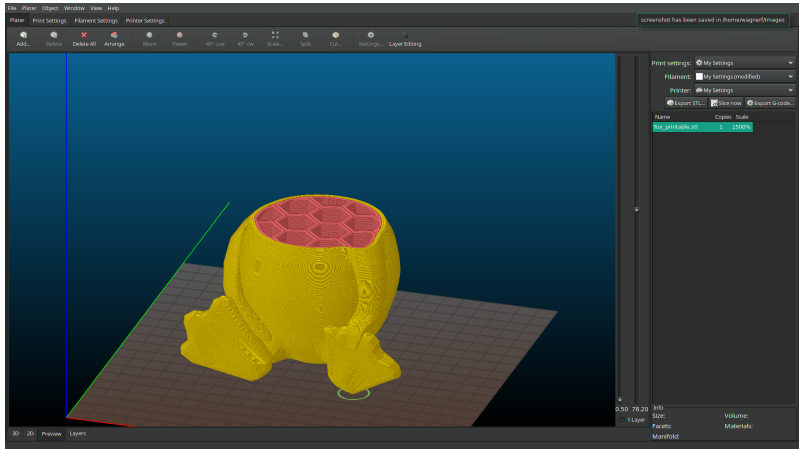
Séance VII – Bentley Ottmann

---

Frédéric Wagner

Ensimag 1<sup>ère</sup> année – 2016-2017





## Bentley Ottmann

Principe

Difficultés

Code fourni

Travail demandé

## Sorted Containers

## Bentley Ottmann

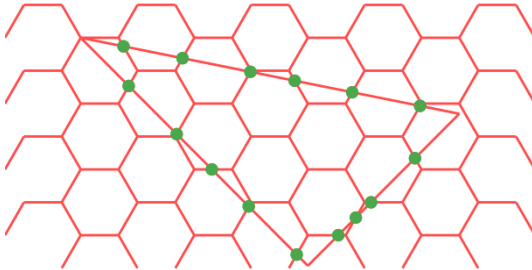
Principe

Difficultés

Code fourni

Travail demandé

## Sorted Containers



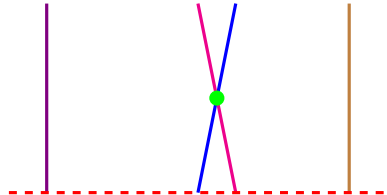
**Entrées :** Ensemble  $S$  de segments, sans overlap.

**Sorties :** Pour chaque  $s \in S$ , un ensemble de points d'intersections.

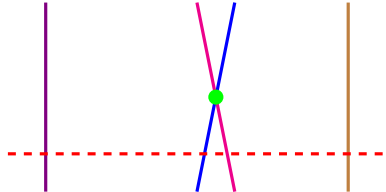
```
1  pour chaque couple  $(s_1, s_2)$  de segments distincts
    faire
2      si  $s_1$  intersecte  $s_2$  en  $i$  alors
3          ajouter  $i$  aux intersections de  $s_1$ ;
4          ajouter  $i$  aux intersections de  $s_2$ ;
5      finsi
6  fin
```

- ▶ coût au pire cas  $\Theta(n^2)$ 
  - ▶ constante plus ou moins grande
- ▶  $10^9 \Rightarrow 10^7 \Rightarrow n < 10^{3.5}$
- ▶ si pavage :  $m^2$  segments
- ▶ pire si l'algorithme est appelé un grand nombre de fois

- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan

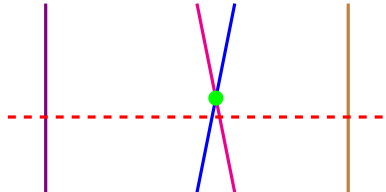


- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan

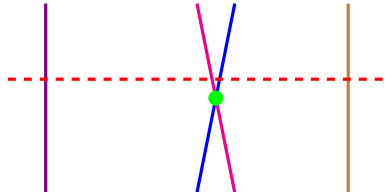




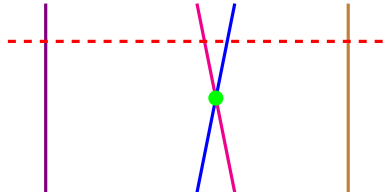
- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



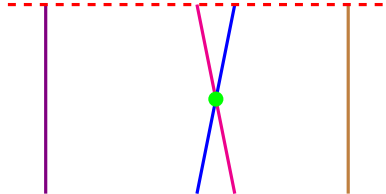
- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



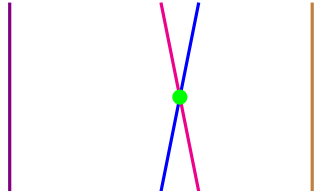
- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



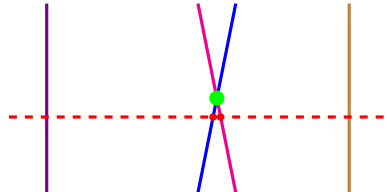
- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



- ▶ similaire à l'algorithme d'intersections de facettes (BPI)
- ▶ simulation d'un balayage du plan



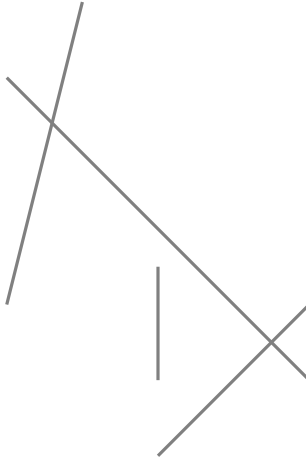
## Idée principale

Juste avant une intersection, les deux segments sont voisins.

- ▶ simulation d'un balayage du plan
  - ▶ seuls les endroits **Intéressants** sont simulés
  - ▶ évènements ( = points ?)
- ▶ on stocke les segments **en vie**
- ▶ lorsqu'un segment apparaît on teste l'intersection avec ses voisins
- ▶ lorsqu'un segment disparaît on teste l'intersection de ses voisins

## Exemple d'exécution

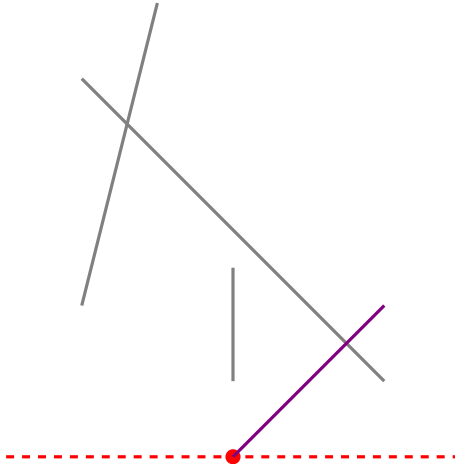
---





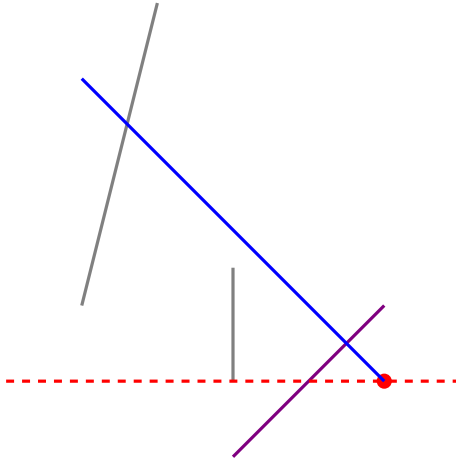
## Exemple d'exécution

---

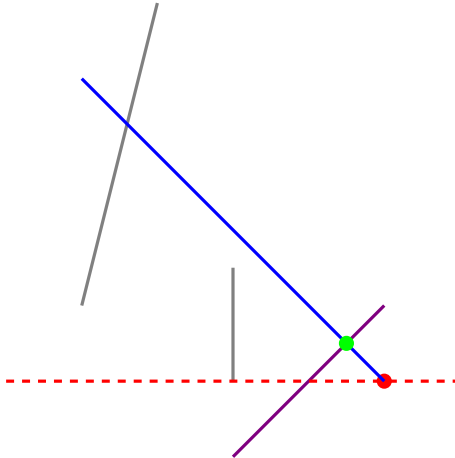


## Exemple d'exécution

---

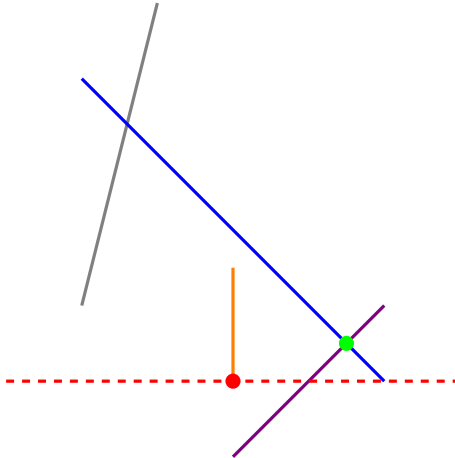


## Exemple d'exécution



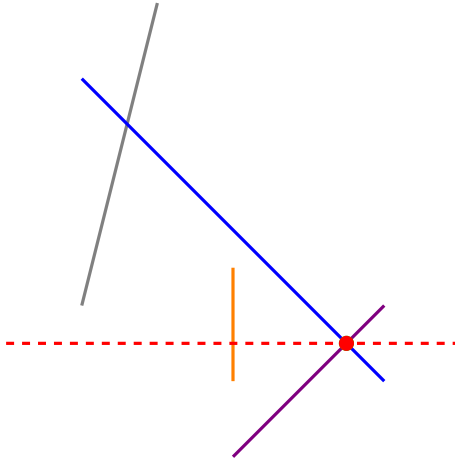
## Exemple d'exécution

---

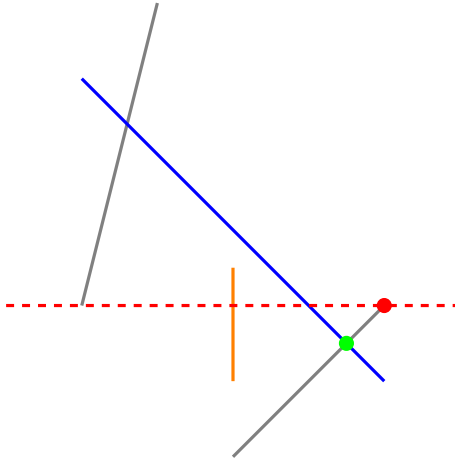


## Exemple d'exécution

---

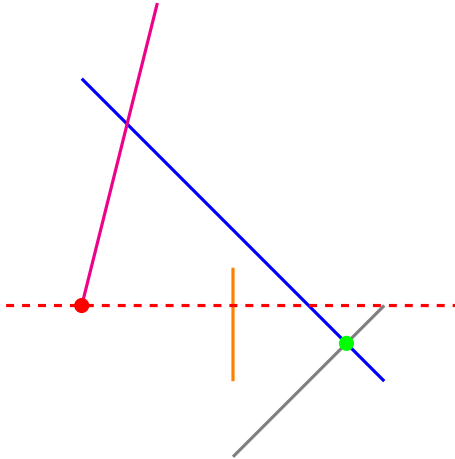


## Exemple d'exécution

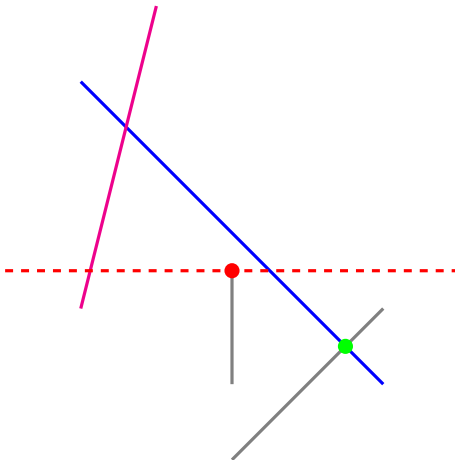


## Exemple d'exécution

---

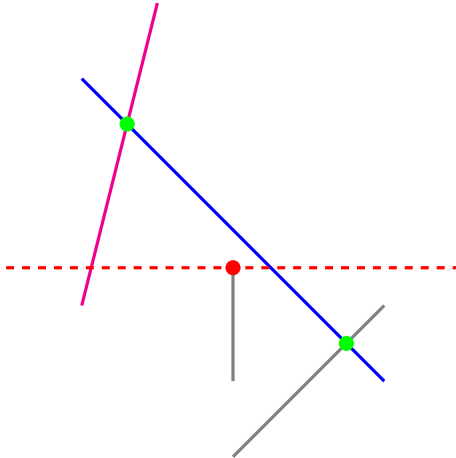


## Exemple d'exécution

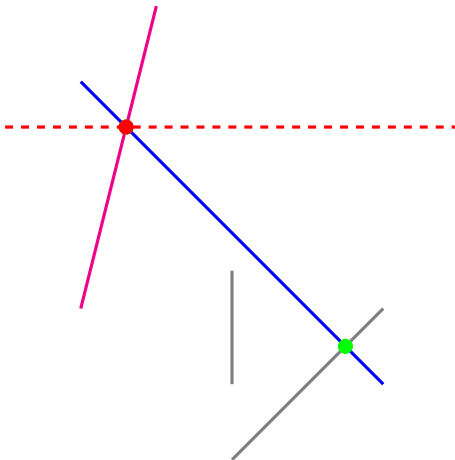




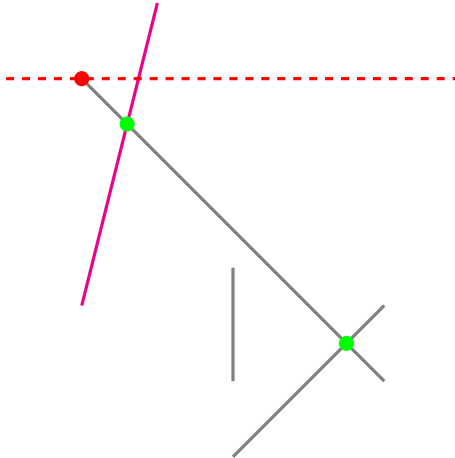
## Exemple d'exécution



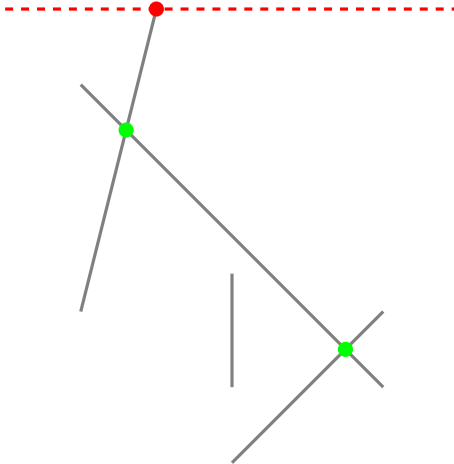
## Exemple d'exécution



## Exemple d'exécution

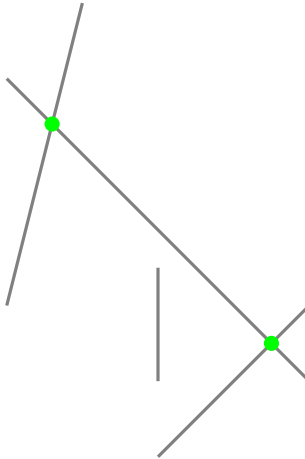


## Exemple d'exécution

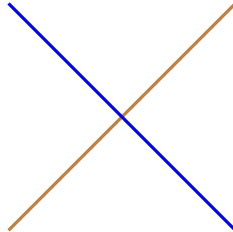


## Exemple d'exécution

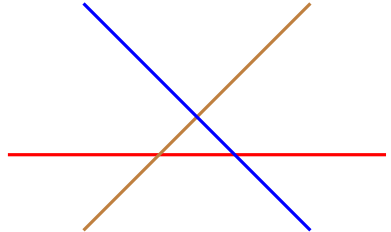
---



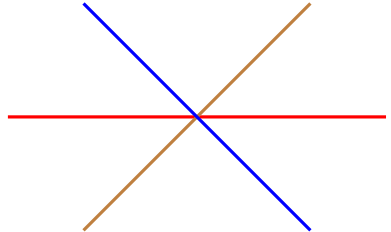
- balayage de bas en haut (par rapport au svg)
- relation à gauche de



- balayage de bas en haut (par rapport au svg)
- relation à gauche de

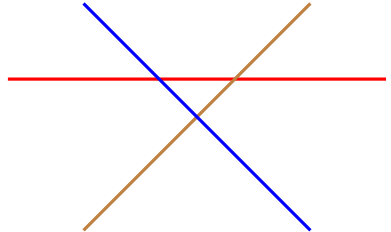


- balayage de bas en haut (par rapport au svg)
- relation à gauche de

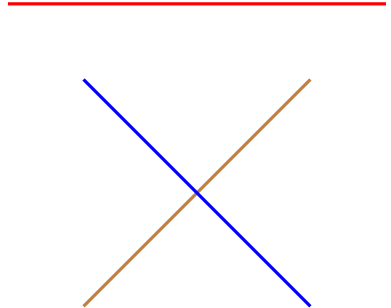




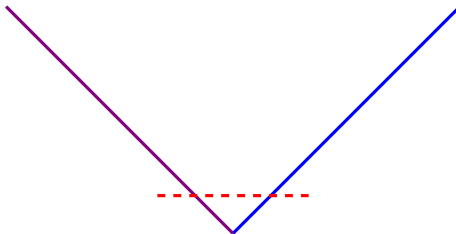
- balayage de bas en haut (par rapport au svg)
- relation à gauche de



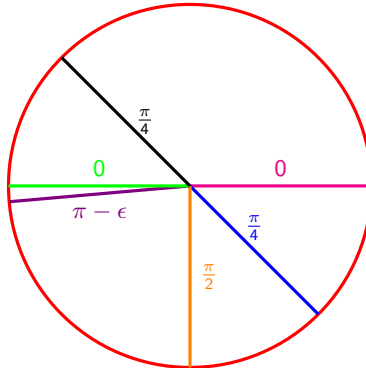
- balayage de bas en haut (par rapport au svg)
- relation à gauche de



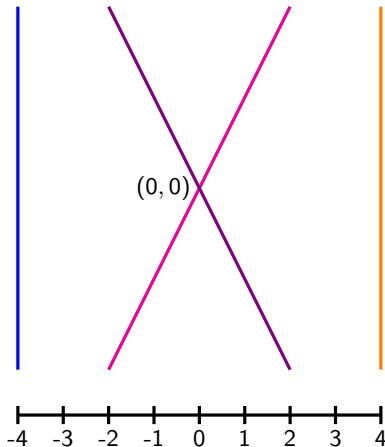
- ▶ objectif : comparaison simple de segments
- ▶ mais :
  - ▶ la comparaison dépend de la position courante
    - ▶ fonction **clef**(segment, position)  $\rightarrow (x, \text{angle})$
  - ▶ pour certaines positions, segments non comparables
    - ▶ ne pas appeler avant ou après le segment
    - ▶ on prend la **convention** suivante : être à une intersection signifie 'juste après' l'intersection



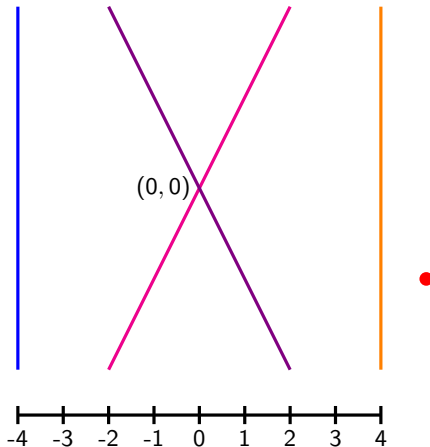
- utilisation d'angles lors des intersections
- support du segment



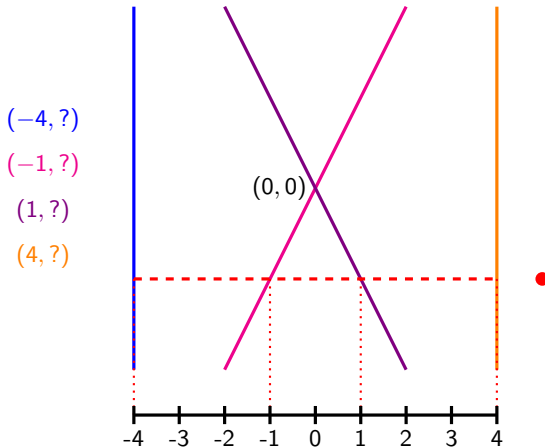
## Clefs : Principe



## Clefs : Principe



## Clefs : Principe



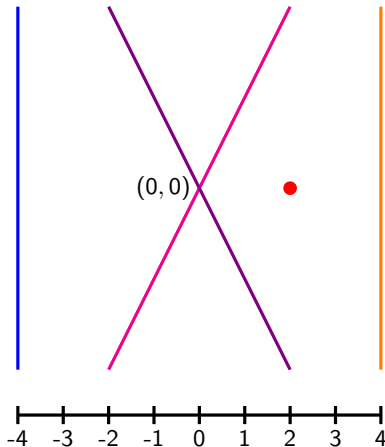
## Clefs : Principe

$$(-4, -\frac{\pi}{2})$$

$$(0, -\frac{5\pi}{8})$$

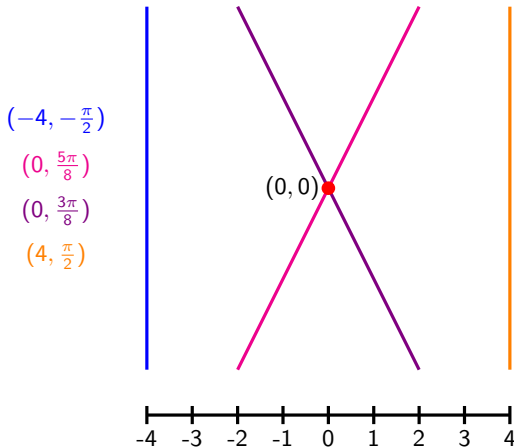
$$(0, -\frac{3\pi}{8})$$

$$(4, \frac{\pi}{2})$$

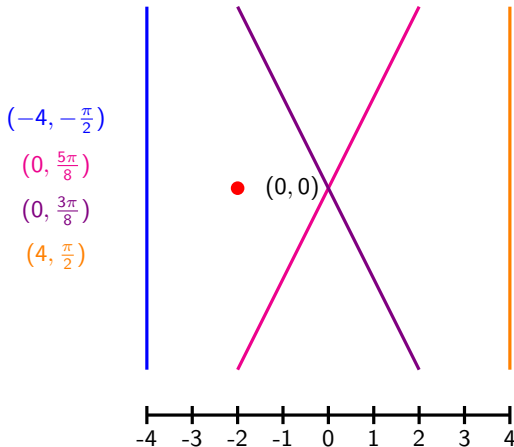




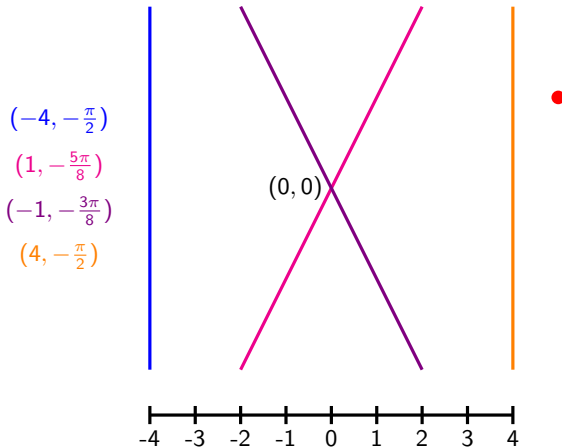
## Clefs : Principe



## Clefs : Principe

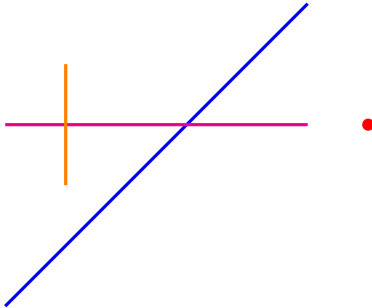


## Clefs : Principe



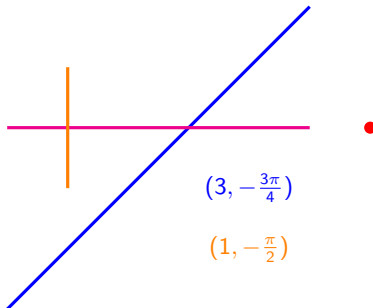
## Cas particulier : segment horizontal

---



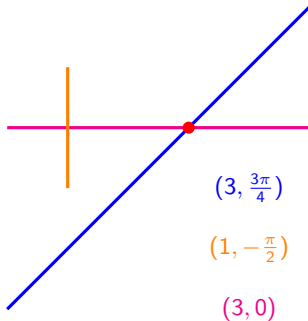
- l'intersection est au point courant

## Cas particulier : segment horizontal



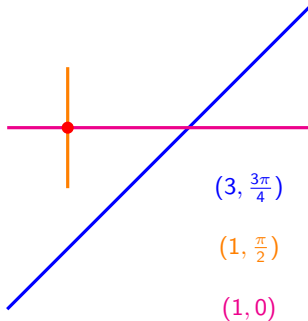
- l'intersection est au point courant

## Cas particulier : segment horizontal



- l'intersection est au point courant

## Cas particulier : segment horizontal



- l'intersection est au point courant

**Entrées :**  $S$ , l'ensemble des segments

**Sorties :** Pour chaque segment, les intersections qui ne tombent pas sur une extrémité

```
1  pour chaque  $s \in S$  faire
2  |   créer les évènements  $E$  de début et de fin
3  fin
4  segments_vivants  $\leftarrow \emptyset$ ;
5  point_courant  $\leftarrow \text{None}$ ;
6  tant que  $E$  n'est pas vide faire
7  |   enlever de  $E$  l'évènement  $e$  le plus proche;
8  |   terminer les segments de  $e$  (et détecter les intersections);
9  |   point_courant  $\leftarrow$  point de  $e$  (pour les clefs de comparaison);
10 |   démarrer les segments de  $e$  (et détecter les intersections);
11 fintq
```



- ▶ lors de la détection :
  - ▶ stocker le résultat
  - ▶ créer des évènements de début et fin de segments

### Filtrer

Une intersection peut être à la fin (ou au début) d'un segment tout en étant à l'intérieur de l'autre.

## Bentley Ottmann

Principe

Difficultés

Code fourni

Travail demandé

## Sorted Containers

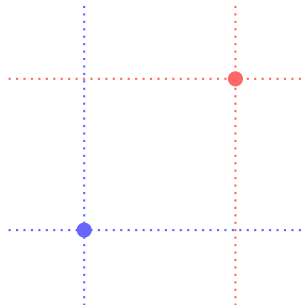
- ▶ de nombreux problèmes d'arrondis sont possibles :
  - ▶ intersections entre droites quasi-horizontales et la droite de balayage
  - ▶  $i$  intersection de  $s_1$   $s_2$  ; il est possible que l'intersection de  $s_1$  avec la droite  $y = i.y$  ne donne pas  $i$
  - ▶ création de points dans le passé
  - ▶ ...

### Exemple

Calcul de la pente d'une droite quasi verticale

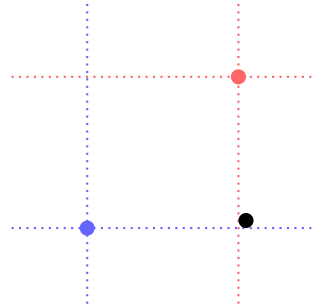
$$p = \frac{y_2 - y_1}{x_2 - x_1}$$

- ▶ on contraint les points pour éviter certains problèmes
- ▶ quasi horizontal  $\Rightarrow$  horizontal
- ▶ quasi vertical  $\Rightarrow$  vertical
- ▶ changements non visibles pour l'utilisateur
- ▶ ajouts en  $\Theta(1)$

**Attention !**

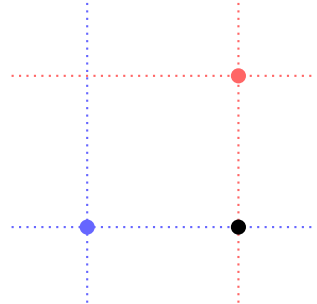
Tout nouveau point doit **obligatoirement** passer par l'ajusteur.

- ▶ on contraint les points pour éviter certains problèmes
- ▶ quasi horizontal  $\Rightarrow$  horizontal
- ▶ quasi vertical  $\Rightarrow$  vertical
- ▶ changements non visibles pour l'utilisateur
- ▶ ajouts en  $\Theta(1)$

**Attention !**

Tout nouveau point doit **obligatoirement** passer par l'ajusteur.

- ▶ on contraint les points pour éviter certains problèmes
- ▶ quasi horizontal  $\Rightarrow$  horizontal
- ▶ quasi vertical  $\Rightarrow$  vertical
- ▶ changements non visibles pour l'utilisateur
- ▶ ajouts en  $\Theta(1)$

**Attention !**

Tout nouveau point doit **obligatoirement** passer par l'ajusteur.

- ▶ calcul des clefs
- ▶ intersection droite horizontale  $\Leftrightarrow$  segments
- ▶ pour certains  $y$  : ne pas faire le calcul car on connaît déjà la réponse
  - ▶ mise en cache
    - ▶ création initiale des évènements
    - ▶ création des évènements d'intersection
  - ▶ calcul de clef : lire d'abord dans le cache

### Attention !

D'abord l'ajusteur.

## Bentley Ottmann

Principe

Difficultés

Code fourni

Travail demandé

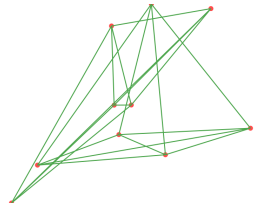
## Sorted Containers



- ▶ petite bibliothèque de géométrie
- ▶ `class Point`
- ▶ `class Segment`
- ▶ intersection de segments
- ▶ affichage graphique
- ▶ chargement de fichiers de tests

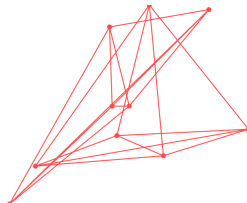
```
points = [  
    [Point([random(), random()]) for _ in range(5)]  
    for _ in range(2)  
]  
segments = [  
    [Segment(endpoints) for endpoints in combinations(p, r=2)]  
    for p in points  
]
```

```
tycat(points, segments)
```



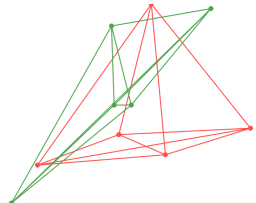
```
points = [  
    [Point([random(), random()]) for _ in range(5)]  
    for _ in range(2)  
]  
segments = [  
    [Segment(endpoints) for endpoints in combinations(p, r=2)]  
    for p in points  
]
```

```
tycat(  
    zip(iter(points), iter(segments))  
)
```



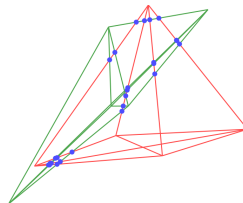
```
points = [  
    [Point([random(), random()]) for _ in range(5)]  
    for _ in range(2)  
]  
segments = [  
    [Segment(endpoints) for endpoints in combinations(p, r=2)]  
    for p in points  
]
```

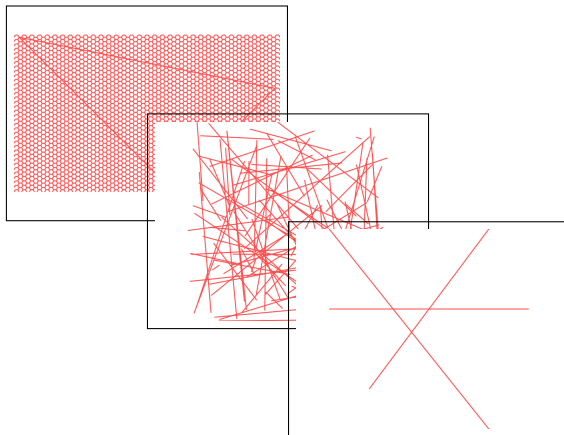
```
tycat(  
    *zip(iter(points), iter(segments))  
)
```



```
points = [  
    [Point([random(), random()]) for _ in range(5)]  
    for _ in range(2)  
]  
segments = [  
    [Segment(endpoints) for endpoints in combinations(p, r=2)]  
    for p in points  
]
```

```
intersections = filter(None, (  
    c[0].intersection_with(c[1])  
    for c in product(*segments)  
))  
tycat(segments[0], segments[1],  
    intersections)
```





► `load_segments`

- ▶ CoordinatesHash
- ▶ insertions en  $O(1)$
- ▶ constante faible pour les points déjà vus
- ▶ auto-utilisé lors du chargement des tests

## Bentley Ottmann

Principe

Difficultés

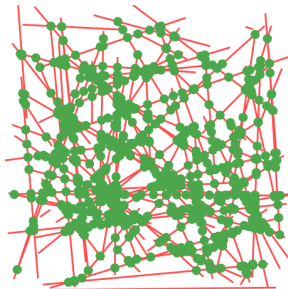
Code fourni

Travail demandé

## Sorted Containers



- ▶ implémenter Bentley Ottmann
- ▶ le code fonctionne au moins sur les tests fournis
  - ▶ `bo.py`
- ▶ mesure de performance (courbe)
- ▶ rapport centré sur les expériences



- ▶ de nombreuses variantes sont possibles pour l'implémentation
- ▶ toutes les idées sont les bienvenues
  - ▶ implémenter plusieurs variantes
  - ▶ peut également être différents choix de paramètres
  - ▶ validation expérimentale
    - ▶ objectif : pouvoir convaincre quelqu'un

### Précisons

Seul l'algorithme et l'utilisation de python pur est imposé. **tout est changeable**

- ▶ l'algorithme est complexe
- ▶ le débogage est encore plus complexe
  - ▶ avancer pas à pas
  - ▶ tester chaque morceau de code
    - ▶ écriture de tests
  - ▶ ne pas cibler immédiatement les exemples complexes
- ▶ échanges d'idées : OK
- ▶ échanges de code : PAS OK

- ▶ biais expérimentaux classiques :
  - ▶ perturbations de la machine
  - ▶ variations aléatoires
  - ▶ comparaisons faisant varier plusieurs paramètres d'un coup
- ▶ présentation
  - ▶ une courbe exhibe un comportement
    - ▶ on choisit la courbe en fonction du comportement que l'on veut montrer
    - ▶ n'hésitez pas à mesurer des nombres d'appels ou temps passés dans différentes fonctions et à le mettre en lien avec ce que l'on attendrait théoriquement

## Bentley Ottmann

Principe

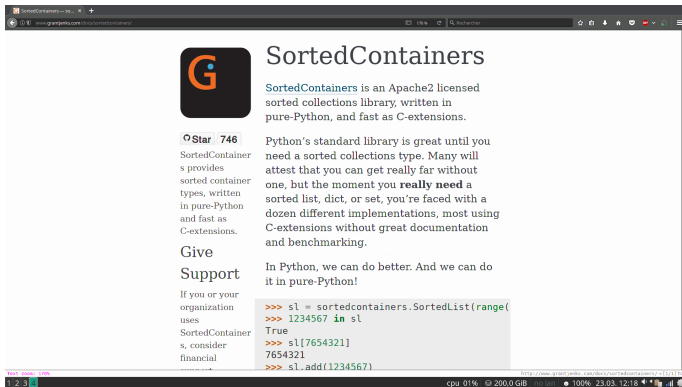
Difficultés

Code fourni

Travail demandé

## Sorted Containers

<http://www.grantjenks.com/docs/sortedcontainers/>



The screenshot shows a web browser displaying the SortedContainers project page. The page features a dark header with the project name 'SortedContainers' and a logo. Below the header, there is a description of the library as an Apache2 licensed sorted collections library written in pure-Python. A sidebar on the left includes a GitHub star count of 746 and a 'Give Support' section. The main content area contains a paragraph about the library's benefits and a code block demonstrating its usage with a SortedList.

**SortedContainers**

SortedContainers is an Apache2 licensed sorted collections library, written in pure-Python, and fast as C-extensions.

**Star 746**

SortedContainers provides sorted container types, written in pure-Python and fast as C-extensions.

**Give Support**

If you or your organization uses SortedContainers, consider financial support.

Python's standard library is great until you need a sorted collections type. Many will attest that you can get really far without one, but the moment you **really need** a sorted list, dict, or set, you're faced with a dozen different implementations, most using C-extensions without great documentation and benchmarking.

In Python, we can do better. And we can do it in pure-Python!

```
>>> sl = sortedcontainers.SortedList(range(
>>> 1234567 in sl
True
>>> sl[7654321]
7654321
>>> sl.add(1234567)
```

## Sorted Container

---

- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés

## Sorted Container

---

- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés
- ▶ utilisation d'un vecteur ?

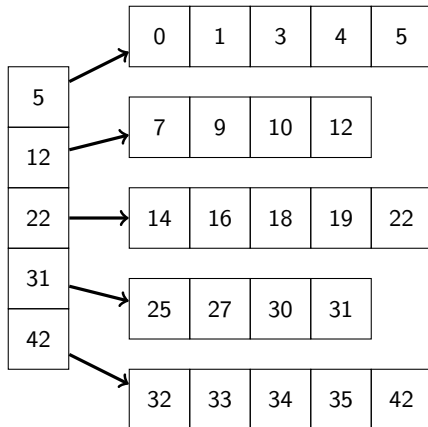


- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés
- ▶ utilisation d'un vecteur ?
- ▶ remove / insert
  - ▶ déplace potentiellement tout le monde
  - ▶ coût :  $\Theta(n)$

- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés
- ▶ utilisation d'un vecteur ?
- ▶ remove / insert
  - ▶ déplace potentiellement tout le monde
  - ▶ coût :  $\Theta(n) = O(1)$

- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés
- ▶ utilisation d'un vecteur ?
- ▶ remove / insert
  - ▶ déplace potentiellement tout le monde
  - ▶ coût :  $\Theta(n) = O(1)$  (pour  $n$  suffisamment petit 😊)

- ▶ objectif : remplacer les ABR
- ▶ ensemble d'éléments ordonnés
- ▶ utilisation d'un vecteur ?
- ▶ remove / insert
  - ▶ déplace potentiellement tout le monde
  - ▶ coût :  $\Theta(n) = O(1)$  (pour  $n$  suffisamment petit 😊)
- ▶ limite sur la taille



- ▶  $n$  éléments
- ▶ limite  $m$
- ▶ chaque vecteur entre  $c \times m$  et  $m$  éléments
- ▶  $O\left(\frac{n}{m}\right)$  vecteurs

- ▶  $n$  éléments
- ▶ limite  $m$
- ▶ chaque vecteur entre  $c \times m$  et  $m$  éléments
- ▶  $O\left(\frac{n}{m}\right)$  vecteurs
- ▶ algorithme de recherche ?

- ▶  $n$  éléments
- ▶ limite  $m$
- ▶ chaque vecteur entre  $c \times m$  et  $m$  éléments
- ▶  $O\left(\frac{n}{m}\right)$  vecteurs
- ▶ algorithme de recherche ?
  - ▶ dichotomie pour trouver le bon vecteur
  - ▶ dichotomie pour trouver la bonne position



- ▶  $n$  éléments
- ▶ limite  $m$
- ▶ chaque vecteur entre  $c \times m$  et  $m$  éléments
- ▶  $O\left(\frac{n}{m}\right)$  vecteurs
- ▶ algorithme de recherche ?
  - ▶ dichotomie pour trouver le bon vecteur
  - ▶ dichotomie pour trouver la bonne position
- ▶ coût ?

- ▶  $n$  éléments
- ▶ limite  $m$
- ▶ chaque vecteur entre  $c \times m$  et  $m$  éléments
- ▶  $O(\frac{n}{m})$  vecteurs
- ▶ algorithme de recherche ?
  - ▶ dichotomie pour trouver le bon vecteur
  - ▶ dichotomie pour trouver la bonne position
- ▶ coût ?

$$O\left(\log\left(\frac{n}{m}\right) + \log(m)\right) = O(\log(n))$$

- ▶ trouver la position  $O(\log(n))$
- ▶ ajouter l'élément dans le vecteur  $O(m)$
- ▶ si besoin :
  - ▶ scinder le vecteur en 2  $O(m)$
  - ▶ insérer le nouveau vecteur dans la table des max  $O(\frac{n}{m})$

- ▶ trouver la position  $O(\log(n))$
- ▶ ajouter l'élément dans le vecteur  $O(m)$
- ▶ si besoin :
  - ▶ scinder le vecteur en 2  $O(m)$
  - ▶ insérer le nouveau vecteur dans la table des max  $O(\frac{n}{m})$
- ▶ de quelle type d'analyse a-t'on besoin ?

- ▶ trouver la position  $O(\log(n))$
- ▶ ajouter l'élément dans le vecteur  $O(m)$
- ▶ si besoin :
  - ▶ scinder le vecteur en 2  $O(m)$
  - ▶ insérer le nouveau vecteur dans la table des max  $O(\frac{n}{m})$
- ▶ de quelle type d'analyse a-t'on besoin ?

### Coût amorti

On moyenne le coût sur des séquences d'opérations

## Coût amorti

---

- ▶ analyse **à la louche**
- ▶ on part d'un conteneur de  $n$  éléments
- ▶ on ajoute  $n$  nouveaux éléments

- ▶ analyse **à la louche**
- ▶ on part d'un conteneur de  $n$  éléments
- ▶ on ajoute  $n$  nouveaux éléments
- ▶ **en gros**  $O(\frac{n}{m})$  nouveaux vecteurs

- ▶ analyse **à la louche**
- ▶ on part d'un conteneur de  $n$  éléments
- ▶ on ajoute  $n$  nouveaux éléments
- ▶ **en gros**  $O(\frac{n}{m})$  nouveaux vecteurs
- ▶ au total  $O(n \log(n) + nm + (\frac{n}{m})(m + \frac{n}{m})) = O(n \log(n) + nm + \frac{n^2}{m^2})$



- ▶ analyse **à la louche**
- ▶ on part d'un conteneur de  $n$  éléments
- ▶ on ajoute  $n$  nouveaux éléments
- ▶ **en gros**  $O(\frac{n}{m})$  nouveaux vecteurs
- ▶ au total  $O(n \log(n) + nm + (\frac{n}{m})(m + \frac{n}{m})) = O(n \log(n) + nm + \frac{n^2}{m^2})$
- ▶ en amorti  $O(\log(n) + m + \frac{n}{m^2})$

- ▶ analyse à la louche
- ▶ on part d'un conteneur de  $n$  éléments
- ▶ on ajoute  $n$  nouveaux éléments
- ▶ en gros  $O(\frac{n}{m})$  nouveaux vecteurs
- ▶ au total  $O(n \log(n) + nm + (\frac{n}{m})(m + \frac{n}{m})) = O(n \log(n) + nm + \frac{n^2}{m^2})$
- ▶ en amorti  $O(\log(n) + m + \frac{n}{m^2})$
- ▶ en choisit la meilleure valeur de  $m$  :
  - ▶  $m = \frac{n}{m^2} \Rightarrow m = \sqrt[3]{n}$

### Coût amorti

$$O\left(\log(n) + \sqrt[3]{n} + \frac{n}{n^{\frac{2}{3}}}\right) = O(\sqrt[3]{n})$$

## Sorted Containers vs ABR

